TP 1
Introduction au projet
Serveur partie 1

Xavier de Rochefort
xderoche@labri.fr
`http://www.labri.fr/~xderoche/RE216/`

10 octobre 2013

1 Le projet

Afin de vous familiariser avec l'utilisation des sockets *POSIX* en C, on vous propose de réaliser les bases d'un grand classique de la programmation réseau : un cas pratique de discussion instantanée de type client / serveur. À titre d'exemple et de curiosité, vous pouvez jeter un coup d'œil au protocole *IRC* (Internet Relay Chat) défini originellement par la RFC1459.

La réalisation du projet se fera en binôme. Vous devrez rendre votre réalisation accompagnée d'un compte-rendu expliquant de manière synthétique le mode d'emploi de votre application, vos choix techniques et les raisons de ces choix, et toutes les informations utiles à l'évaluation de votre travail.

Les bases du projet seront guidées au travers de 4 séances encadrées, suite auxquelles il vous sera proposée une série d'améliorations à réaliser en autonomie durant le mois suivant la fin des TP. La date exacte de rendue vous sera communiquée par mail courant novembre.

2 Première séance

2.1 Objectif

Lors des TDs d'introduction à la programmation réseau, vous avez été amenés à développer un serveur *echo* multi-clients utilisant la primitive *select*. Nous allons partir du code du serveur *echo* implémenté en TD pour obtenir la base de notre serveur de chat.
Pour éviter d'avoir à implémenter une partie cliente lors de ce premier TP, vous pouvez utiliser les commandes *telnet* ou *nc* (utilisez l'option -h des commandes pour en savoir plus sur leur fonctionnement)

2.2 Prérequis

Le TP nécessite d'avoir terminé la base de code d'un serveur *echo* multi-clients décrit dans la feuille de TD 2. (`https://www.labri.fr/perso/bromberg/cours/ENSEIRB/RE205/TDs/td2_reseaux_2012.pdf`). Le serveur doit accepter la connexion de plusieurs clients et renvoyer les chaines de caractères reçues au client qui les a envoyées.

3 Questions

Q1.1 Faites évoluer le code initial du serveur *echo* pour que les caractères reçus soit envoyés à tous les clients sauf à l'émetteur (*broadcast*).

Aide : Pensez en premier lieu à ajouter de quoi gérer les clients connectés : les stocker avec leurs informations (ex : numéro de socket), les parcourir, les ajouter, les supprimer...Rappelez vous vos cours de C de l'an passé sur les structures de données, les tableaux et les listes simplement/doublement chainées etc.

Q1.2 Adaptez la phase de connexion d'un client pour récupérer son pseudo. N'oubliez pas de gérer le cas particulier des doublons.

Q1.3 Permettez à un client de pouvoir récupérer la liste des pseudos des autres clients après envoi d'une commande particulière au serveur.

Un client doit pouvoir envoyer un message à un seul de ses pairs (unicast) en indiquant le pseudo en guise d'identifiant du destinataire.

Q1.4 Permettez à un client d'envoyer une chaine de caractère à un autre client en indiquant le pseudo du destinataire.

Bonus

Un utilisateur peut avoir envie de parler à une sous partie des pairs connectés (*multicast*). IRC par exemple propose la notion de *salon* permettant à plusieurs personnes connaissant l'identifiant du salon de s'y rattacher et de voir tous les messages des personnes de ce même salon.

Q1.5 Permettez aux clients l'envoi de messages multicast.

Note sur l'erreur `bind(): address already in use`

Ajouter l'option `SO_REUSEADDR` à la socket permet d'éviter l'erreur `bind(): address already in use` (code ci-après).

```
int optval = 1;
setsockopt(socket, SOL_SOCKET, SO_REUSEADDR, &optval, sizeof(optval));
```